Практическая работа №18, ПРОГРАММИРУЕМЫЙ ТЕСТ

1. Постановка задачи

Написать программу тестирования, которая удовлетворяет следующим требованиям:

1. Количество вопросов теста может варьировать от 5 до 20.
2. Для каждого вопроса должно быть предусмотрено четыре варианта ответов.
3. В результате тестирования должна быть выставлена оценка (неудовлетворительно, удовлетворительно, хорошо, отлично) в зависимости от количества правильных ответов.
4. Вопросы и ответы теста должны находиться в файле.
5. В программе должна быть заблокирована возможность возврата к предыдущему вопросу.

Новыми в этой программе являются:

- использование типизированных файлов,
- построение форм в ходе выполнения программы,
- использование компонента **SpinEdit** (редактор числа) со страницы палитры компонентов **Samples**.

2. Информационная постановка задачи

В условии задачи сказано, что для хранения используется файл. В Delphi поддерживает три разных подхода для работы с файлами, для разного типа информации желательно использовать наиболее подходящий подход.

2.1. Немного теории

Текстовые файлы – в файле находится текстовая информация (набор строк из символов).
Текстовая информация состоит не обязательно из букв, в ней могут содержаться любые символы из таблицы ASCII.

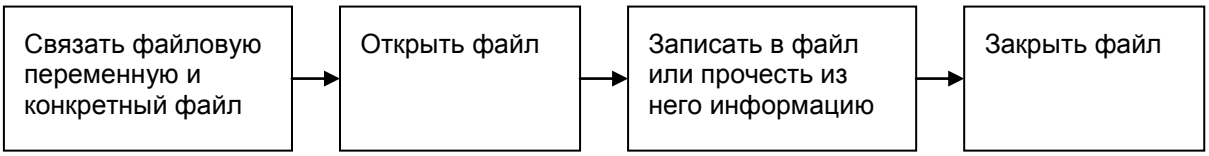
Типизированные файлы – в файле находится информация любого рода. Но структура такой информации обязательно должна повторяться. Название типизированный файл получил из-за того, что в нем хранится однотипная информация. Одна строка типа называется записью типизированного файла.

> **Примечание**
>
> Типизированные файлы очень часто используются для описания каких-либо списков информации, например, телефонных справочников. Каждая запись данного списка может состоять, например, из номера телефон абонента, имя, адрес.

Нетипизированные файлы – файл содержит последовательность байт без какой-либо структуры. Например, это может быть закодированный или сжатый блок информации.

При работе с таким файлом вся информация рассматривается как набор отдельных кусков по N байт, где значение N задается программистом в диапазоне от 1 до 65535.

В процессе работы для любого файла нужно выполнять последовательность следующих действий:



Менять местами пункты нельзя.

Для каждого файла есть понятие – текущее положение в файле, т.е. это то положение, в котором на данный момент идет обработка информации. Например, в типизированном файле мы можем прочесть первую запись, затем вторую и так далее. В таком случае текущим положением в файле сначала будет первая запись, потом вторая, затем третья.

2.2. Структура информации

В нашей задаче единицей информации будет являться вопрос, который в свою очередь будет состоять из текста вопроса, четырех вариантов ответа и четырех указателей правильности ответа. Таким образом, можно ввести тип, описывающий вопрос.

```
Type TTEST = Record
     TEXT      : String [250];// Текст Вопроса
     OTV_1     : String [100];// Вариант ответа № 1
     OTV_2     : String [100];// Вариант ответа № 2
     OTV_3     : String [100];// Вариант ответа № 3
     OTV_4     : String [100];// Вариант ответа № 4
     REZ_1     : Byte         // Признак правильности ответа № 1
     REZ_2     : Byte         // Признак правильности ответа № 2
     REZ_3     : Byte         // Признак правильности ответа № 3
     REZ_4     : Byte         // Признак правильности ответа № 4
        end;
```

TEXT, **OTV_1**, **OTV_2**, **OTV_3**, **OTV_4** содержат текстовую информацию, а **REZ_1**, **REZ_2**, **REZ_3**, **REZ_4** будут содержать числовую информацию: 0 – ответ неправильный, 1 – ответ правильный. Можно было бы для «Признака правильности ответа» ввести тип *Boolean*, но мы ввели числовое значение, т.к. в ходе выполнения программы будем суммировать признаки правильности для ответов, которые выбирает тестируемый. В результате мы получим количество правильных ответов.

В предложенную структуру можно внести изменения, которые позволят в дальнейшем сократить текст программы. Для этого определим тип массива для данных «Вариант ответа» и «Признак правильности ответа». Тогда описание типа будет выглядеть:

```
Type TTEST = Record
     TEXT      : String [250];               // Текст Вопроса
     OTV       : Array [1..4]of String [100];// Варианты ответов
     REZ       : Array [1..4]of Byte         // Правильный ответ
        end;
```

Непроизвольно мы подошли к тому, что при работе будем использовать типизированный файл.

Для работы программы тестирования нам будет нужна еще информация о названии теста и количестве вопросов в тесте. Количество вопросов в тесте (количество записей в файле) можно узнать с помощью функции **FileSize**. А вот название теста необходимо поместить в файл. Поэтому информация в файле будет располагаться следующим образом:

Первая запись содержит название теста, все последующие записи – информацию о вопросах. В результате количество записей в файле будет на одну больше, чем количество вопросов в тесте.

Для формирования теста мы будем использовать массив:

```
Const KOL = 21;                          // Max количество вопросов
Var   TEST_: Array [1..KOL] of TTEST;  // Массив ТЕСТА
```

2.3. Описание файла

При объявлении файловой переменной типизированных файлов указывается тип данных, содержащихся в файле.

```
Var FFile: File of TTEST;                 // Описание файла
```

Для чтения или записи файлов такого типа возможно использование только операторы **Read** и **Write**. **ReadLn** и **WriteLn** использовать нельзя, т.к. данные в файле это не строки текста, т.е. нет и линий.

Данные в типизированном файле хранятся во внутреннем машинном виде. Поэтом создавать файл и читать информацию из него можно только с помощью программы, которая «знает» структуру данных. Для этого мы создадим две программ. Первая будет отвечать за создание теста, а вторая – тестирование.

3. Программа создание теста

Откройте новый проект и разместите в форме компоненты в соответствии с рис.1.

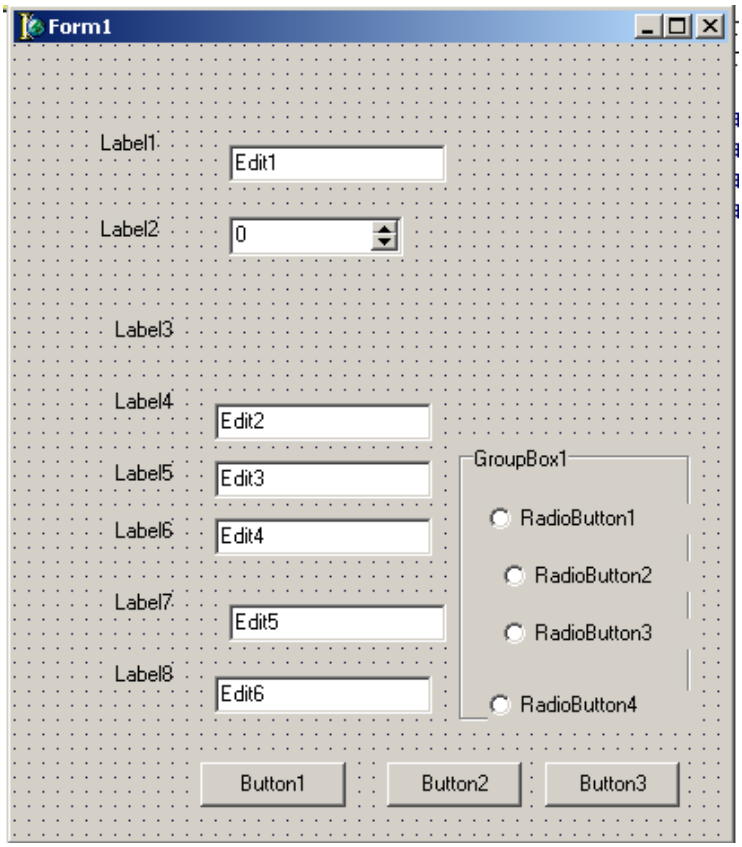


Рис.1

Нам понадобятся следующие компоненты:

№ п/п	Наименование компонента	Страница	Для чего предназначен
1	Label1	Standard	Вывод текста «Название теста»
2	Label2	Standard	Вывод текста «Количество вопросов» в тесте
3	Label3	Standard	Вывод текста «Вопрос №»
4	Label4	Standard	Вывод текста «Текст» вопроса
5	Label5	Standard	Вывод текста «Ответ № 1»
6	Label6	Standard	Вывод текста «Ответ № 2»
7	Label7	Standard	Вывод текста «Ответ № 3»
8	Label8	Standard	Вывод текста «Ответ № 4»
9	Edit1	Standard	Окно ввода названия теста
10	Edit2	Standard	Окно ввода текста вопроса
11	Edit3	Standard	Окно ввода варианта ответа № 1
12	Edit4	Standard	Окно ввода варианта ответа № 2
13	Edit5	Standard	Окно ввода варианта ответа № 3
14	Edit6	Standard	Окно ввода варианта ответа № 4
15	SpinEdit1	Samples	Выбор количества вопросов в *Тесте*. **Примечание** *SpinEdit* – редактор числа. Обеспечивает отображение и редактирование целого числа. Этот компонент содержит две кнопки со стрелками, используемые для увеличения или уменьшения значения числа. Диапазон значений числа задается свойствами *Min* и *Max*. Эти свойства определяют, соответственно, минимально и максимально возможные значения числа. Шаг изменения значения при нажатии кнопок со стрелками содержится в свойстве *Increment*. По умолчанию значение этого свойства равно единице. Текущая значение определяется свойства *Text*.
16	GroupBox1	Standard	Панель *GroupBox* позволяет объединить компоненты *RadioButton* для того, чтобы кнопки взаимодействовали друг с другом в группе (может быть нажата только одна кнопка из четырех).
17	RadioButton1	Standard	Выбор правильного ответа
18	RadioButton2	Standard	
19	RadioButton3	Standard	
20	RadioButton4	Standard	
21	Button1	Standard	Кнопка перехода на предыдущий вопрос
22	Button2	Standard	Кнопка перехода на следующий вопрос
23	Button3	Standard	Запись готового теста на диск (заполнены все вопросы)

Сохраните проект. В последующем сохраняйте проект и проверяйте его работоспособность после каждого изменения.

Не стремитесь располагать компоненты на свои места. Это мы сделаем в программе при создании формы. Для этого выделите объект **Form1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **OnCreat**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

```
Var Left_N : Integer;  // Отступ слева верхней части
    Top_N  : Integer;  // Отступ сверху
    Left_NN: Integer;  // Отступ слева для раздела ответов
    Top_NN : Integer;  // Отступ сверху для RadioButton
    K, I   :Integer;
 . . .
Left_N:=30;
Top_N:=50;
Left_NN:=60;

// Формирование элементов формы

Form1.Width:=740;
Form1.Height:=540;
Form1.Caption:='Создание теста';

Label1.Left:=Left_N;
Label1.Top:=Top_N;
Label1.Font.Style:=[fsBold];
Label1.Font.Size:=10;
Label1.Caption:='Название теста';

Edit1.Text:='';
Edit1.Top:=Top_N;
Edit1.Left:=Left_N+Label1.Width+10;
Edit1.Width:=300;

Top_N:=Top_N+40;

Label2.Left:=Left_N;
Label2.Top:=Top_N;
Label2.Font.Style:=[fsBold];
Label2.Font.Size:=10;
Label2.Caption:='Количество вопросов';

SpinEdit1.Left:=Left_N+Label2.Width+10;
SpinEdit1.Top:=Top_N;
SpinEdit1.MinValue:=5;
SpinEdit1.MaxValue:=20;
SpinEdit1.Text:='5';
SpinEdit1.Width:=40;

Top_N:=Top_N+60;

Label3.Left:=Left_N;
Label3.Top:=Top_N;
Label3.Font.Style:=[fsBold];
Label3.Font.Size:=10;
Label3.Caption:='Вопрос № 1';

Top_N:=Top_N+40;

Label4.Left:=Left_NN;
```

```
Label4.Top:=Top_N;
Label4.Font.Style:=[fsBold];
Label4.Font.Size:=9;
Label4.Caption:='Текст';

Edit2.Text:='';
Edit2.Top:=Top_N;
Edit2.Left:=Left_NN+Label4.Width+10;
Edit2.Width:=600;
Left_NN:=Left_NN+Label4.Width+10;

Top_N:=Top_N+40;

GroupBox1.Left:=620;
GroupBox1.Top:=Top_N+20;
GroupBox1.Width:=90;
GroupBox1.Height:=180;
GroupBox1.Caption:=' Правильный ';
GroupBox1.Font.Size:=8;
```

Мы установили свойства компонентам **Label1**, **Edit1**, **Label2**, **SpinEdit1**, **Label3**, **Label4**, **Edit2**, **GroupBox1**. Все они обладают разными свойствами. Но компоненты, которые описывают 4 варианта ответов, можно сгруппировать – **Label** + **Edit** + **RadioButton** (для каждого ответа). Компоненты каждой строки будут отличаться только значением свойства **Top** и своими номерами. Компоненты распределятся по строкам следующим образом:
1-ая строка - Label5, Edit3, RadioButton1
2-ая строка – Label6, Edit4, RadioButton2
3-ая строка – Label7, Edit5, RadioButton3
4-ая строка – Label8, Edit6, RadioButton4
Поэтому возникает вопрос: «Каким образом можно перечислять в программе имена компонентов последовательно, например, переходя от **Label5** к **Label6**, потом к **Label7**?»

Это можно сделать, если воспользоваться методом **FindComponent**. Он возвращает указатель экземпляра компоненты, о которой известно. Допустим, что форма содержит экземпляр компоненты **TLabel** с именем **Label2**. Чтобы получить указатель на экземпляр **Label2** и изменить свойство **Caption**, можно записать:

```
K:=2;
TLabel(FindComponent('Label'+IntToStr(K))).Caption:= 'МЕТКА';
```

Следовательно, для формирования строк вариантов ответов в программу мы можем добавить текст:

```
//  Формирование раздела ОТВЕТОВ
K:=2; Top_NN:= -20;
For I:=1 To 4 do
  begin
  Top_N:=Top_N+40; Top_NN:=Top_NN+40;

  TLabel(FindComponent('Label'+IntToStr(K+I+2))).Left:=Left_NN;
  TLabel(FindComponent('Label'+IntToStr(K+I+2))).Top:=Top_N;
  TLabel(FindComponent('Label'+IntToStr(K+I+2))).Font.Size:=9;
  TLabel(FindComponent('Label'+IntToStr(K+I+2))).Caption:='Ответ № '
+IntToStr(I);
```

```
  TEdit(FindComponent('Edit'+IntToStr(K+I))).Text:='';
  TEdit(FindComponent('Edit'+IntToStr(K+I))).Top:=Top_N;
  TEdit(FindComponent('Edit'+IntToStr(K+I))).Left:=200;
  TEdit(FindComponent('Edit'+IntToStr(K+I))).Width:=400;

 TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Left:=40;

TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Top:=Top_NN;

TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Caption:='';
 TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Width:=10;

  end;
```

И в конце добавляем блок формирования трех кнопок.

```
//  Формирование нижней части формы

Top_N:=Top_N+70;

Button1.Caption:='Предыдущая';
Button1.Left:=Left_NN+100;
Button1.Top:=Top_N;
Button1.Width:=100;
Button1.Height:=30;
Button1.Enabled:=False;

Button2.Caption:='Следующая';
Button2.Left:=Left_NN+250;
Button2.Top:=Top_N;
Button2.Width:=100;
Button2.Height:=30;

Button3.Caption:='Записать';
Button3.Left:=Left_NN+500;
Button3.Top:=Top_N;
Button3.Width:=100;
Button3.Height:=30;
Button3.Enabled:=False;

STEP:=1;   //Номер вопроса = 1
```

В результате, после запуска проекта, вы увидите форму, представленную на рис.2.

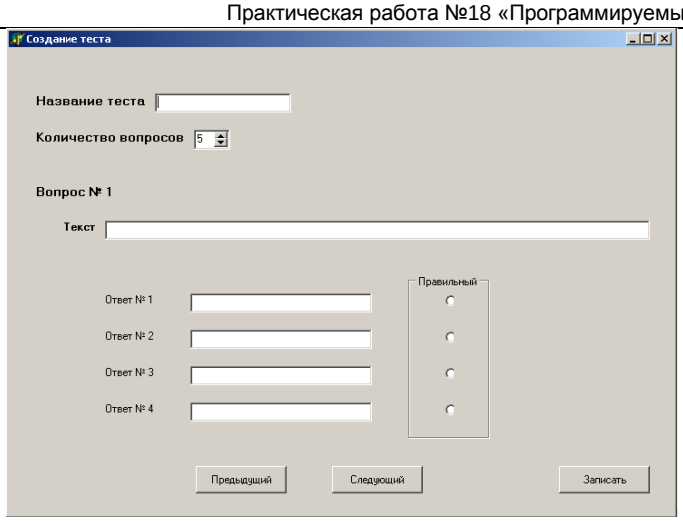


Рис.2

Наша задача – сформировать файл тестовых вопросов, который потом можно использовать для проведения тестирования. Для этого необходимо подумать о том, как будут организованы данные. Из общего вида **Form1** можно сделать вывод, что каждый вопрос состоит из текста вопроса, четырех вариантов ответов и четырех вариантов признаков правильности ответа. Для удобства воспользуемся типом данных **Запись** (**Record**). С одной стороны, запись можно рассматривать как единое целое, с другой стороны – как набор отдельных элементов разного типа. Для нашей задачи подойдет следующее описание типа:

```
Type TTEST = Record
     TEXT      : String [250];                // Текст Вопроса
     OTV       : Array [1..4]of String [40]; // Варианты ответов
     REZ       : Array [1..4]of Byte         // Правильный ответ
        end;
```

Решить нашу задачу можно следующим образом:

1. Ввести название теста и количество вопросов.
2. Ввести все вопросы с вариантами ответов, записывая их в массив.

   Примечание.
   Это позволит просматривать (вперед, назад) вопросы, вносить изменения до записи их в файл.

3. Сохранить все сформированные вопросы в файле.

Для этого наш массив будет иметь тип записи **TTEST**. В раздел описания включим описание файла и описание массива.

```
Const KOL = 21;                          // Max количество
вопросов
Var FFile: File of TTEST;                // Описание файла
    TEST_: Array [1..KOL] of TTEST;      // Массив ТЕСТА
```

При построении массива первый элемент массива (**TEST_[1]**) будет содержать информацию о тесте: **TEST_[1].TEXT** – название теста
Начиная со второго элемента массива, будет располагаться непосредственно информация о вопросах теста.
В процедуру **TForm1.FormCreate** необходимо добавить еще один блок подготовки массива **TEST_** к работе.

```
For I:=1 to KOL do
  begin
  TEST_[I].TEXT:='';
  for K:=1 to 4 do
    begin
    TEST_[I].OTV[K]:='';
    TEST_[I].REZ[K]:=0
    end;
  end;
```

Создадим процедуру, которая обрабатывает ситуацию нажатия кнопки **Button3** (запись созданного массива в файл). Для этого выделите объект **Button3**, перейдите на **вкладку Events Инспектора объектов** (**Object Inspector**), найдите событие **On Click**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

```
procedure TForm1.Button3Click(Sender: TObject);
Var I    :Byte;
    STROK:TTEST;
begin
AssignFile(FFILE,'TEST.txt');
Rewrite(FFILE);
Button1.Enabled:=False;   //Недоступны все три кнопки
Button2.Enabled:=False;
Button3.Enabled:=False;

 For I:=1 to  STEP_N+1 do
  begin
   STROK:=TEST_[I];
   Write(FFILE, STROK)
  end;
 CloseFile(FFILE)
end;
```

В представленной процедуре файловой переменной **FFILE** ставится в соответствие имя файла, т.е. массив мы запишем в файл **TEST.txt**. Затем открывается файл (в нашем случае он создается) для записи. В файл переписываются все элементы массива **TEST_**. Их количество соответствует введенному параметру «Количество вопросов в тесте» плюс 1, т.к. добавлен один элемент (первый), который содержит название теста.

Устанавливаются недоступными кнопки «Предыдущая», «Следующая», «Запись», а затем массив записывается в файл. В конце файл закрывается.

Перейдем к созданию процедуры обработки **Button2** («Следующая» – переход к следующей записи). Для этого выделите объект **Button2**, перейдите на вкладку **Events Инспектора объекто*в*** (**Object Inspector**), найдите событие **On Click**, справа от него дважды щелкнуть левой кнопкой мыши. Разберем код процедуры по частям.

Часть № 1

```
procedure TForm1.Button2Click(Sender: TObject);
Var I:Byte;
begin
If (RadioButton1.Checked=False) and
   (RadioButton2.Checked=False) and
   (RadioButton3.Checked=False) and
   (RadioButton4.Checked=False)
Then
 begin
 ShowMessage('Не выбран правильный ответ');
 Exit
 end;
```

В начале процедуры необходимо проверить выбран ли правильный ответ (должна быть нажата одна из кнопок **RadioButton**). Если значение свойства **Checked у RadioButton** равно **False**, то будет выдано сообщение «Не выбран правильный ответ» и процедура закончит свою работу.

Часть № 2

```
If STEP=1 Then
 begin
  TEST_[1].TEXT:=Edit1.Text;
  Edit1.Enabled:=False;
  STEP_N:= StrtoInt(SpinEdit1.Text);  // Количество вопросов в
тесте
  SpinEdit1.Enabled:=False;
  STEP:=STEP+1;
 end;
```

Этот кусок процедуры необходим, т.к. в массиве элементы с индексом 1 содержат информацию о названии теста. Поэтому, если у нас первый вопрос (**STEP**=1), то осуществляется запись в массив информации о тесте и устанавливаются объекты **Edit1** и **SpinEdit1** недоступными. Переходим к формированию вопросов.

Часть № 3

```
TEST_[STEP].TEXT:=Edit2.Text;

For I:=1 to 4 do
 begin
TEST_[STEP].OTV[I]:=TEdit(FindComponent('Edit'+IntToStr(I+2))).Text;
If TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Checked= True
Then TEST_[STEP].REZ[I]:= 1;
 end;
```

В третьей части элементам массива с индексом **STEP** присваивается значение соответствующих полей. Если нажата **RadioButton**, то соответствующему элементу массива **REZ[I]** (*Результат ответа*) присваивается значение 1.

Часть № 4

```
Button1.Enabled:=True;
If STEP>STEP_N Then
 begin
  Button2.Enabled:=False;
  Button3.Enabled:=True;
 end
Else
 begin
 STEP:=STEP+1;
 Label3.Caption:='Вопрос № '+IntToStr(STEP-1);
 Edit2.Text:=TEST_[STEP].TEXT;
 For I:=1 to 4 do
 begin
TEdit(FindComponent('Edit'+IntToStr(I+2))).Text:=TEST_[STEP].OTV[I];
  If TEST_[STEP].REZ[I]= 1 Then
   TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Checked:= True
   else
   TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Checked:= False
 end
 end
end;
```

В заключительной части сначала устанавливается доступной кнопка **Button1** («Предыдущая»), т.к. уже есть одна запись и мы, после выполнения процедуры, переходим на вторую. Дальше идет проверка – является ли наша запись последняя, т.е. номер текущей записи (**STEP**) больше количества вопросов в тесте (**STEP_N**). Если мы записали в массив все вопросы, то кнопка **Button2** («Следующая») устанавливается недоступной, а кнопка **Button3** («Запись») – доступной.

1. Если не достигли конца теста, то идет подготовка к следующему вопросу: увеличивается переменная **STEP** и свойствам компонент экрана присваиваются значения следующей записи массива.
2. Если массив не заполнен до конца, то это будут пустые записи.
3. Если массив был заполнен, и мы переходили от одной записи к другой с помощью кнопок **Button1** («Предыдущая») и **Button2** («Следующая»), то на экран будут выведены значения соответствующей записи.

Перейдем к созданию процедуры обработки **Button1** («Предыдущая» – переход к предыдущей записи). Для этого выделите объект **Button1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **On Click**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

```
procedure TForm1.Button1Click(Sender: TObject);
Var I: Byte;
begin
STEP:=STEP-1;
Label3.Caption:= 'Вопрос № '+IntToStr(STEP-1);
Edit2.Text:=TEST_[STEP].TEXT;
For I:=1 to 4 do
 begin
```

```
TEdit(FindComponent('Edit'+IntToStr(I+2))).Text:=TEST_[STEP].OTV[I];
If TEST_[STEP].REZ[I]= 1 Then
   TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Checked:= True
   Else
   TRadioButton(FindComponent('RadioButton'+IntToStr(I))).Checked:= False
 end;
If STEP=2 then
 Button1.Enabled:=False;
Button2.Enabled:=True;
end;
```

В начале переходим к предыдущему вопросу (уменьшаем значение переменной **STEP**), а затем выводим на экран содержимое вопроса. В конце процедуры проверяем, если вопрос является первым (номер строки массива **STEP**=2), то делаем недоступной кнопку *Предыдущая* и в любом случае должна быть доступна кнопка *Следующая*.

Сохраните проект окончательно, запустите и протестируйте его.

Задание

Внесите изменение в программу так, чтобы можно было бы просматривать уже созданные тесты.

4. Программа тестирования

Программа тестирования будет состоять из двух форм: титульной формы, на которой осуществляется выбор теста из справочника тестов (текстовый файл) и форма формы непосредственно теста.
В зависимости от правильности ответов на вопросы теста, подсчитывается результат и выставляется оценка.

4.1. Создание титульной формы

Титульная форма будет выполнять следующие функции:

- Поиск справочника тестов (текстовый файл **STEST.txt**), чтение информации о тестах-файлах и вывод перечня тестов в компонент **RadioGroup1**.
- В зависимости от выбранного тест-файла, формирование рабочего файла тестов.
- Вызов формы непосредственного тестирования.
- Удаление рабочего файла тестов при окончании работы программы.

Справочник тестов – текстовый файл, который можно создать в любом текстовом редакторе. Каждая строка файла содержит наименование теста. Может иметь, например, такое содержание:

```
Информация и информационные процессы
Архитектура компьютера
Измерение информации
Основы алгебры логики
```

Каждому тесту соответствует файл-тест, который имеет имя **TEST_N.txt**, где **N** – номер строки в файле.

Откройте новый проект и разместите в форме компоненты в соответствии с рис.3.

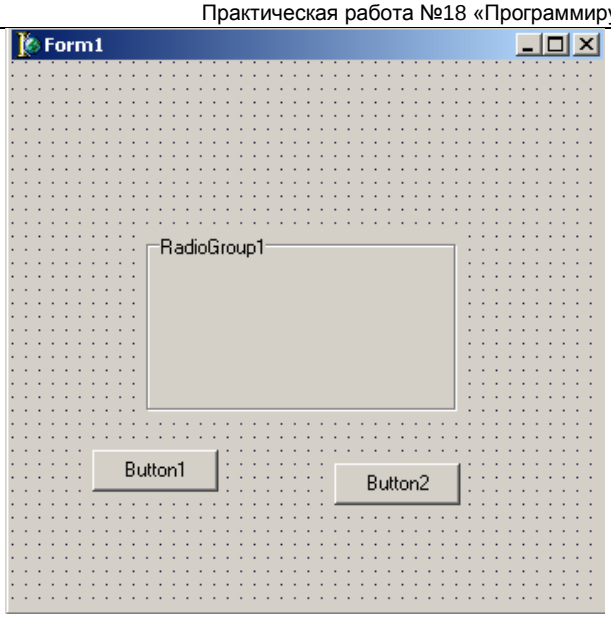


Рис.3

№ п/п	Наименование компонента	Страница	Для чего предназначен
1	RadioGroup1	Standard	Перечень возможных тестов (из справочника тестов)
2	Button1	Standard	Кнопка для перехода к тестированию после выбора теста
3	Button2	Standard	Кнопка выхода из программы тестирования

В разделе **implementation** разместите описание типа и данных, которые мы будем использовать при работе.

```
Type TTEST = Record
     TEXT      : String [250];                // Текст вопроса
     OTV       : Array [1..4]of String [40]; // Варианты ответов
     REZ       : Array [1..4]of Byte         // Правильный ответ
        end;
Var SFILE : TextFile;     // Справочник тестов
    FFile: File of TTEST; // Файл тестов
    FF_R : File of TTEST; // Рабочий файл тестов
    KOL  : Byte;          // Количество тестов в справочнике
тестов
    TEST_: TTEST;         // Строка записи файла
```

Выделите объект **Form1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **OnCreat**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

```
procedure TForm1.FormCreate(Sender: TObject);
Var I      : Byte;
    SF     : String;
    T_TEXT : Array [1..10] of String[50];
    Left_N : Integer;    // отступ слева
    Top_N  : Integer;    // отступ справа
begin
```

```
KOL:=1;
AssignFile(SFILE,'STEST.txt');
{$I-} Reset(SFILE); {$I+}
If IOResult = 0 then
 begin
  Repeat
    ReadLn(SFILE,SF);
    T_TEXT[KOL]:=SF;
    KOL:=KOL+1;
  until (Eof(SFILE)) or (KOL>10);
  CloseFile(SFILE);

  Left_N:=30;
  Top_N:=50;

  Form1.Caption:='Тестирование';
  Form1.Width:=400;
  Form1.Height:=Top_N+ KOL*25 + 120;

  RadioGroup1.Top:=Top_N;
  RadioGroup1.Left:=Left_N;
  RadioGroup1.Width:= 350;
  RadioGroup1.Height:= KOL*25;
  RadioGroup1.Caption:= ' Выбери тест ';
  For I:=1 to KOL-1 do
   RadioGroup1.Items.Add(T_TEXT[I]);
  RadioGroup1.Font.Size:=11;

  Button1.Caption:='Тест';
  Button1.Left:= Left_N;
  Button1.Top:=Top_N+ KOL*25+40;
  Button1.Width:= 100;
  Button1.Height:=30;

  Button2.Caption:='Выход';
  Button2.Left:= Left_N+ 250;
  Button2.Top:=Top_N+ KOL*25+40;
  Button2.Width:= 100;
  Button2.Height:=30;
 end
  else
   ShowMessage('Отсутствует справочник тестов');
end;
```

Немного теории

Директива компилятора {$I+}/{$I-} включает или выключает автоматическую проверку ошибок ввода/вывода. Когда директива включена и возникает ошибка ввода/вывода, то выполнение программы завершается с сообщением о произошедшей ошибке. Но можно отключить аварийное завершение программы при возникновении ошибки. В этом случае программист берет на себя обработку аварийной ситуации. Для того, чтобы узнать произошла ли ошибка при открытии файла, можно воспользоваться функцией *IOResult*. Она возвращает код отличный от нуля, если произошла ошибка и ноль, если операция выполнилась успешно.

Типичной ошибкой ввода/вывода является отсутствие файла. По умолчанию директива компилятора включена – {$I+}.

В начале процедуры открывается файл справочника тестов **STEST.txt**.

Если файл существует, то информация из него переписывается в массив **T_TEXT** и подсчитывается количество записей (**KOL**).

В программе установлено ограничение на количество элементов массива **T_TEXT** – не более 10. По желанию это можно изменить. После устанавливаются свойства компоненты форма.

Список значений свойства **Items** компоненты **RadioGroup1** будем формировать в ходе программы в зависимости от количества элементов в массиве справочника тестов. Для этого к свойству **Items** применяем метод **Add**:

```
For I:=1 to 4 do RadioGroup1.Items.Add(T_TEXT[I]);
```

Если файл справочника тестов отсутствует, то будет выдано сообщение *Отсутствует справочник тестов.*

В результате форма будет выглядеть так же как на рис.4.

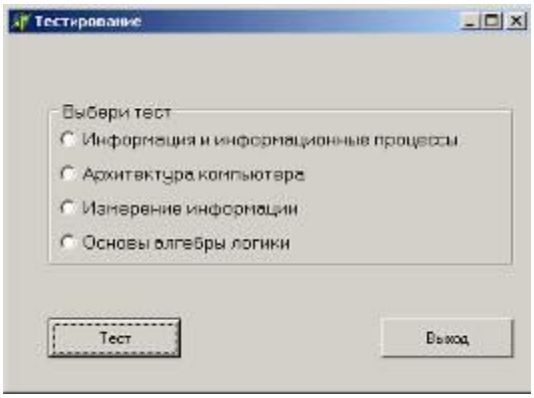


Рис.4

Создайте новую форму **Form2** и сохраните под именем **Unit2.pas**. Подсоедините созданную форму к титульной форме. Для этого в **Unit1.pas** в разделе **interface** в строку списка подсоединенных модулей добавьте **Form2**.

Создадим процедуру, которая обрабатывает ситуацию нажатия кнопки **Button1** (*Тест*). Для этого выделите объект **Button1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **On Click**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

```
procedure TForm1.Button1Click(Sender: TObject);
Var N:Byte;
begin
N:=RadioGroup1.ItemIndex+1;
AssignFile(FFILE,'TEST_'+IntToStr(N)+'.txt'); // Тест с номером
N
{$I-} Reset(FFILE); {$I+}
If IOResult = 0 then
 begin
  AssignFile(FF_R, 'TEST_R.txt');  // Рабочий файл тестов
  Rewrite(FF_R);
  Repeat
    Read(FFILE,TEST_);
```

```
    Write(FF_R, TEST_);
  until (Eof(FFILE));
  CloseFile(FFILE);
  CloseFile(FF_R);
  Form2.ShowModal;
 end
 else ShowMessage('Отсутствует файл тестов');

end;
```

В начале процедуры формируется имя теста в зависимости от номера выбранной строки компонента **RadioGroup1**.

> **Примечание**
>
> Индекс первого переключателя равен 0, а массив справочника содержит индекс первого элемента равный 1.

Дальше открывается выбранный файл-тест. Если данный файл отсутствует, то выдается сообщение «Отсутствует файл тестов». Если при открытии файла-теста не возникло ошибки, то информация из него, переписывается в рабочий файл тестов (**TEST_R.txt**). В завершении открывается форма как модальная. Это означает, что управление передается новой форме, и пользователь не может передать фокус другой форме данного приложения до тех пор, пока он не закроет модальную форму.

Создадим процедуру, которая обрабатывает ситуацию нажатия кнопки **Button2** («Выход»). Для этого выделите объект **Button2**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **On Click**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Close;
end;
```

4.2. Создание формы тестирования

Сделайте активной **Form2** и разместите на ней компоненты в соответствии с рис.5.

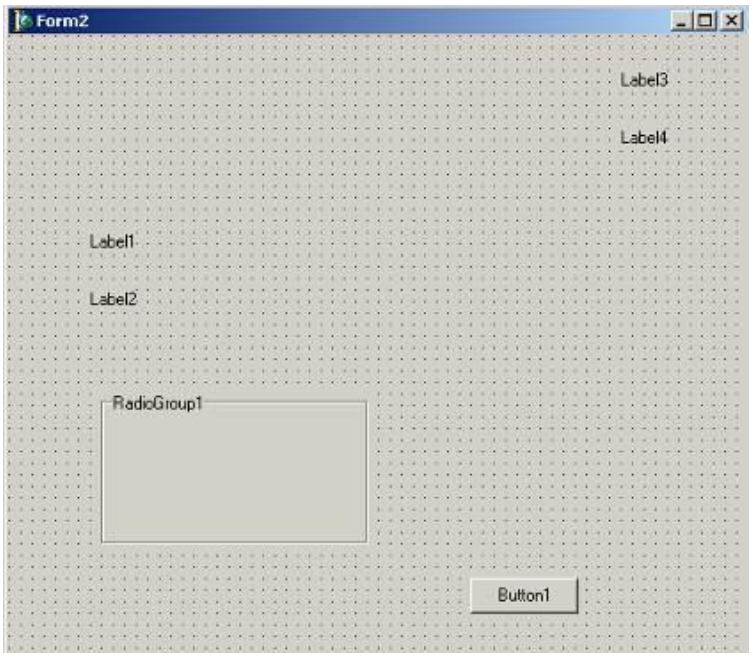


Рис.5

№ п/п	Наименование компонента	Страница	Для чего предназначен
1	Label1	Standard	Вывод текста «Вопрос № …»
2	Label2	Standard	Вывод текста «Текст» вопроса
3	Label3	Standard	Вывод текста «Всего вопросов …»
4	Label4	Standard	Вывод текста «Правильных ответов …»
5	RadioGroup1	Standard	Вывод вариантов ответов
6	Button1	Standard	Кнопка для перехода к следующему вопросу, показу результата и завершения тестирования

В разделе **implementation** разместите описание типа и данных, которые мы будем использовать при работе.

```
Type TTEST = Record
     TEXT      : String [250];                // Текст вопроса
     OTV       : Array [1..4]of String [100];// Варианты ответов
     REZ       : Array [1..4]of Byte          // Правильный ответ
        end;
Var FF_R : File of TTEST; // Рабочий файл тестов
    KOL  : Byte;          // Количество вопросов в тесте
    KOL_N: Byte;          // Текущий номер вопроса
    TEST_: TTEST;         // Строка записи файла
    REZ_N: Byte;          // Количество правильных ответов
```

Для объекта **Form2**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **On Activate**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

```
procedure TForm2.FormActivate(Sender: TObject);
Var I : Byte;
begin
AssignFile(FF_R,'TEST_R.txt');
Reset(FF_R);
Read(FF_R,TEST_);
KOL:= FileSize(FF_R)-1;
KOL_N:=1;
REZ_N:=0;

Label1.Visible:=True;
RadioGroup1.Visible:=True;

Form2.Caption:=TEST_.TEXT;
Form2.Height:=420;
Form2.Width:=850;

Label3.Caption:='Всего вопросов - '+IntToStr(KOL);
Label3.Top:=20;
Label3.Left:= 700;
Label3.Font.Size := 8;

Label4.Caption:='Правильных ответов - '+IntToStr(REZ_N);
Label4.Top:=40;
Label4.Left:= 700;
```

```
Label4.Font.Size := 8;
Label4.Font.Color:=clBlack;

Button1.Caption:='Следующий';
Button1.Top:=350;
Button1.Left:=720;
Button1.Width:= 100;
Button1.Height:=30;

Read(FF_R,TEST_);

Label1.Caption:='Вопрос № '+IntToStr(KOL_N);
Label1.Font.Style := [fsBold];
Label1.Font.Size := 11;
Label1.Height:=50;
Label1.Width:=400;
Label1.Left:=40;
Label1.Top:=70;

Label2.Caption:=TEST_.TEXT;
Label2.Font.Size := 12;
Label2.Height:=50;
Label2.Width:=500;
Label2.Left:=60;
Label2.Top:=100;
Label2.WordWrap := True;
Label2.AutoSize := False;

RadioGroup1.Top:=160;
RadioGroup1.Left:=60;
RadioGroup1.Width:= 750;
RadioGroup1.Height:= 150;
RadioGroup1.Font.Size:=11;
RadioGroup1.Caption:= ' Выбери ответ ';
RadioGroup1.Items.Clear;
 For I:=1 to 4 do
  RadioGroup1.Items.Add(TEST_.OTV[I]);

end;
```

Работа данной процедуры начинается с того, что открывается рабочий файл тестов **TEST_R.txt** и считывается первая запись, которая содержит название теста. С помощью функции **FileSize** определяем количество записей в файле, а количество вопросов будет на единицу меньше.

Дальше настраиваем компоненты **Form2**, **Label3**, **Label4**, **Button1**. Читаем из рабочего файла тестов еще одну запись – первый вопрос. После этого настраиваем компоненты **Label1**, **Label2**, **RadioGroup1**. При описании компонента **RadioGroup1** мы добавляем

```
RadioGroup1.Items.Clear;
```

Эта строка позволяет нам очистить список значений свойства **Items** компоненты **RadioGroup1**, который формируется в ходе программы. Применение метода **Clear** нам необходимо при повторном тестировании – очистить список от предыдущих вариантов ответов.

В результате форма будет выглядеть так же как на рис.6.

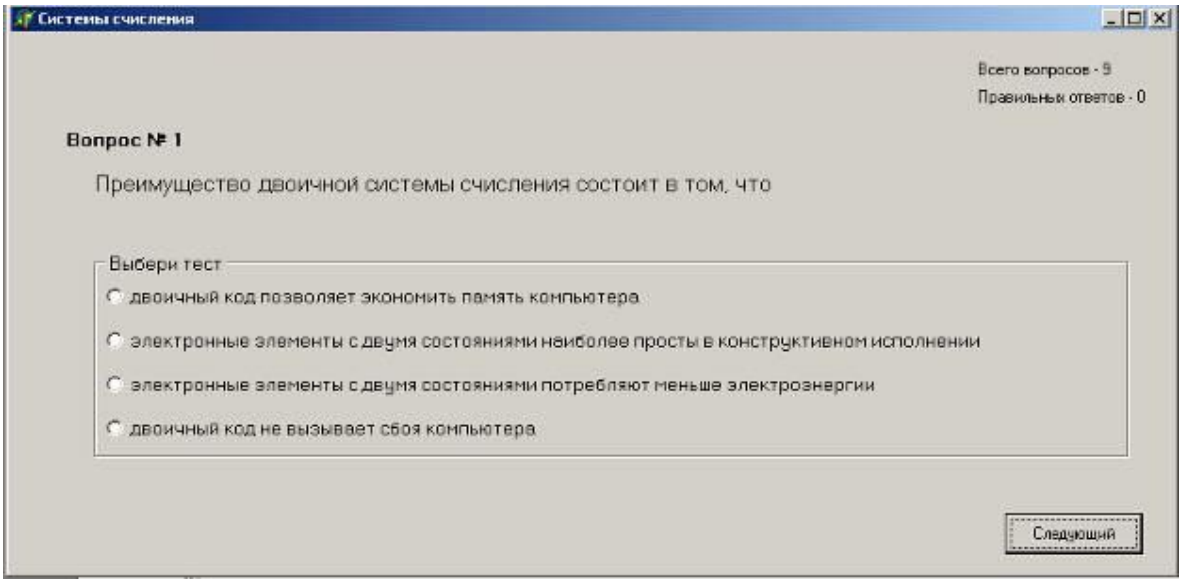


Рис.6

Создадим процедуру, которая обрабатывает ситуацию нажатия кнопки **Button1**. Для этого выделите объект **Button1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **On Click**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий код:

Часть № 1

```
procedure TForm2.Button1Click(Sender: TObject);
Var N, I   :Byte;
    OZENKA : Byte;
begin

If Button1.Caption= 'Результат' Then
 begin
  OZENKA:=Round(REZ_N/KOL*5);
  Button1.Caption:= 'Выход';
  Label1.Visible:=False;
  RadioGroup1.Visible:=False;

  Label2.Font.Size := 12;
  Label2.Left:=250;
  Label2.Top:=150;
  Label2.Caption:='Количество правильных ответов - '+IntToStr(REZ_N)+
  ' из '+IntToStr(KOL);

  Label3.Font.Size := 14;
  Label3.Font.Style := [fsBold];
  Label3.Left:=300;
  Label3.Top:=200;
  Label3.Caption:='Оценка';

  Label4.Font.Size := 14;
  Label4.Left:=350;
  Label4.Top:=200;
  Case OZENKA of
```

```
  1..2: Label4.Font.Color:=clPurple;
  3   : Label4.Font.Color:=clMaroon;
  4   : Label4.Font.Color:=clGreen;
  5   : Label4.Font.Color:=clNavy;
  end;
  Label4.Caption:=IntToStr(OZENKA);
 end
 else
 If Button1.Caption= 'Выход' Then
  begin
  Close;
  CloseFile(FF_R);
  end
 else
 begin
  N:=RadioGroup1.ItemIndex+1;
  REZ_N:=REZ_N+ TEST_.REZ[N];
  KOL_N:=KOL_N+1;

  If KOL_N>KOL Then
  begin
   Label4.Caption:='Правильных ответов - '+IntToStr(REZ_N);
   Button1.Caption:= 'Результат'
  end
   else
    begin
     Label4.Caption:='Правильных ответов - '+IntToStr(REZ_N);
     Read(FF_R,TEST_);
     Label1.Caption:='Вопрос № '+IntToStr(KOL_N);
     Label2.Caption:=TEST_.TEXT;
     RadioGroup1.Items.Clear;
     For I:=1 to 4 do
      RadioGroup1.Items.Add(TEST_.OTV[I]);
    end;
  end;
end;
```

В ходе программы кнопка **Button1** может иметь надписи «Следующая», «Результат», «Выход».

Если кнопка имеет надпись «Результат» и вы кликнули на ней, то в этом случае определяется значение переменной **OZENKA** по пяти бальной системе в зависимости от количества правильных ответов **REZ_N** и общего количества вопросов **KOL**. Затем устанавливается надпись кнопки «Выход», формируется новое изображение формы, выводится итоговая оценка, делаются невидимыми компоненты **Label1** и **RadioGroup1** (поэтому в процедуре **TForm2.FormActivate** устанавливаются данные компоненты видимыми).

Если кнопка имеет надпись «Выход» и вы кликнули на ней, то в этом случае закрывается рабочий файл тестов и закрывается форма.

Если кнопка имеет надпись «Следующий» и вы кликнули на ней, то в этом случае определяется какой был выбран ответ на вопрос, результат складывается с переменной **REZ_N**. Увеличивает значение переменной **KOL_N** (номер текущего вопроса). Если новый текущий номер больше количества вопросов, то изменяется надпись на кнопке

(«Результат»), в противном случае переходим к чтению нового вопроса из рабочего файла тестов и формируем вывод нового вопроса.

Все, программа готова. Осталось только ее протестировать.

Задание для самостоятельного выполнения

	Задание
1	Написать программу, которая осуществляет конвертирование информации из текстового файла в типизированный файл, для файла с тестовыми заданиями.
2	Предусмотреть возможность выбора случайным образом тестового задания из общего списка заданий.
3	Создать файл, в котором накапливается информация о прохождении тестирования. Информация, которая может находиться в нем следующая: фамилия, имя, класс, номер вопрос, номер ответа учащегося, количество правильных ответов, количество неправильных ответов, время начала тестирования, время окончания тестирования и т.п.